Amendments to the Claims:

This listing of the claims will replace all prior versions, and listings, of claims in the application:

Listing of the Claims:

1. (Original) A method of operating a file server in a data network, said method comprising:

(a) the file server receiving a request for metadata about a file to be accessed, the request being received from a data processing device in the data network; and

(b) in response to the request for metadata, the file server granting to the data processing device a lock on at least a portion of the file, and returning to the data processing device metadata of the file including information specifying data storage locations in the file server for storing data of the file.

2. (Original) The method as claimed in claim 1, wherein the file server includes a data storage device including the data storage locations, and a data mover computer for managing locks on files having data stored in said data storage device, wherein the data storage device stores metadata of a plurality of files having file data stored in the data storage device, the data mover computer is coupled to the data storage device for transfer of the metadata between the data storage device and the data mover computer, the data mover computer has a random access memory, and the method includes the data mover computer maintaining a metadata cache in the random access memory, and the method includes the data mover computer accessing the metadata cache for obtaining the metadata that is returned to the data processing device.

3. (Original) The method as claimed in claim 1, wherein a plurality of data processing devices in the data network share read-write access to the file, and the file server grants respective read locks and write locks to the data processing devices in the data network.

4. (Original) The method as claimed in claim 1, wherein the data processing device writes data to the data storage locations in the file server, modifies the metadata from the file server in accordance with the data storage locations in the file server to which the data is written, and sends the modified metadata to the file server.

5. (Original) The method as claimed in claim 4, wherein the data processing device sends the modified metadata to the file server after the data processing device writes the data to the data storage of the file server.

6. (Original) The method as claimed in claim 1, wherein the data processing device has a cache memory for caching the metadata of the file including a version identifier associated with the metadata of the file, and wherein the data processing device includes the version identifier in the request for access to the file, the file server compares the version identifier from the data processing device to a version identifier of a most recent version of the metadata of the file, and the file server returns the most recent version of the metadata of the file to the data processing device when the comparison of the version identifier from the data processing device to the version identifier of the most recent version of the metadata of the file indicates that the metadata of the file cached in the cache memory of the data processing device is not the most recent metadata of the file.

7. (Original) The method as claimed in claim 6, wherein the version identifier is a number that is incremented when the metadata of the file is modified.

8. (Original) A method of operating a file server and a client in a data network, said method comprising:

(a) the client sending to the file server at least one request for access to a file;

(b) the file server receiving said at least one request for access to the file, granting to the client a lock on at least a portion of the file, and sending to the client metadata of the file including information specifying data storage locations in the file server for storing data of the file;

(c) the client receiving from the file server the metadata of the file, using the metadata of the file to produce at least one data access command for accessing the data storage locations in the file server, and sending the data access command to the file server to access the data storage locations in the file server; and

(d) the file server responding to the data access command by accessing the data storage locations in the file server.

9. (Original) The method as claimed in claim 8, wherein the file server includes a data storage device including the data storage locations, and a data mover computer for managing locks on files having data stored in said data storage device, and wherein the client sends to the data mover computer said at least one request for access to the file, the data mover computer responds to said at least one request for access to the file by returning to the client the metadata of the file, and wherein the client sends the data access command to the data storage device over a data transmission path that bypasses the data mover computer.

10. (Original) The method as claimed in claim 9, wherein the data storage device stores metadata of a plurality of files having file data stored in the data storage device, the data mover computer is coupled to the data storage device for transfer of the metadata between the data storage device and the data mover computer, the data mover computer has a random access memory, and the method includes the data mover computer maintaining a metadata cache in the random access memory, and the method includes the data mover computer accessing the metadata cache for obtaining the metadata that is sent to the client.

11. (Original) The method as claimed in claim 8, wherein a plurality of clients in the data network share read-write access to the file, and the file server grants respective read locks and write locks to the clients in the data network.

12. (Original) The method as claimed in claim 8, wherein the lock on at least a portion of the file granted by the file server to the client is not granted to any particular application process of the client, and wherein the client has a lock manager that grants a local file lock to a particular application process that accesses the file.

13. (Original) The method as claimed in claim 8, wherein the client has a lock manager that responds to a request from an application process of the client for access to the file by granting to the application process a local file lock on at least a portion of the file, and then sending to the file server said at least one request for access to the file.

14. (Original) The method as claimed in claim 8, wherein the method includes dynamically linking application programs of the client with input-output related operating system routines of the client, the input-output related operating system routines intercepting file access calls from client application processes to send file access requests to the file server to obtain from the file server locks upon at least a portion of each of the files, to obtain metadata for producing data access commands for accessing data storage in the file server, to produce the data access commands from the metadata, and to send the data access commands to the file server in order to access the data storage of the file server.

15. (Original) The method as claimed in claim 8, wherein the data access command is a write command for a write operation upon at least a portion of the file, and wherein the method includes the client writing the data to the data storage locations in the file server, modifying the metadata from the file server in accordance with the write operation upon at least a portion of the file, and sending the modified metadata to the file server.

16. (Original) The method as claimed in claim 15, wherein the client sends the modified metadata to the file server after the client writes the data to the data storage of the file server.

17. (Original) The method as claimed in claim 16, wherein the client performs asynchronous write operations upon the data storage locations of the file server, and wherein the client sends the modified metadata to the file server in response to a commit request from an application process of the client.

18. (Original) The method as claimed in claim 16, wherein the client performs asynchronous write operations upon the data storage locations of the file server, and wherein the client sends the modified metadata to the file server when the client requests the file server to close the file.

19. (Original) The method as claimed in claim 8, wherein the client has a cache memory for caching the metadata of the file including a version identifier associated with the metadata of the file, and wherein the client includes the version identifier in the request for access to the file, the file server compares the version identifier from the client to a version identifier of a most recent version of the metadata of the file, and the file server returns the most recent version of the metadata of the file to the client when the comparison of the version identifier from the client to the version identifier of the most recent version of the metadata of the file indicates that the metadata of the file cached in the cache memory of the client is not the most recent metadata of the file.

20. (Original) The method as claimed in claim 19, wherein the version identifier is a number that is incremented when the metadata of the file is modified.

21. (Original) A file server comprising:

at least one data storage device for storing a file system; and

a data mover computer coupled to the data storage device for exchange of metadata of files in the file system, the data mover computer having at least one network port for exchange of control information and metadata of files in the file system with data processing devices in the data network, the control information including metadata requests;

wherein the data storage device has at least one network port for exchange of data with the data processing devices in the data network over at least one data path that bypasses the data mover computer; and

wherein the data mover computer is programmed for responding to each metadata request for metadata of a file from each data processing device by granting to said each data processing device a lock on at least a portion of the file, and returning to said each data processing device metadata of the file including information specifying data storage locations in the data storage device for storing data of the file.

22. (Original) The file server as claimed in claim 21, wherein the data mover computer is programmed to receive modified metadata from said each data processing device, and write the modified metadata to the data storage device.

23. (Original) The file server as claimed in claim 21, wherein the data mover computer has a random access memory, and the data mover computer is programmed for maintaining a metadata cache in the random access memory, and the data mover computer is programmed for accessing the metadata cache for obtaining the metadata that is returned to said each data processing device.

24. (Original) The file server as claimed in claim 23, wherein the data mover computer is programmed for receiving modified metadata from said each data processing device, and writing the modified metadata to the metadata cache in the random access memory.

25. (Currently Amended) The file server as claimed in claim 21, wherein the data mover computer is programmed for receiving a metadata version identifier in ~~the~~ <u>said each</u> metadata request<u>, for comparing the metadata version identifier in said each metadata request</u> to a version identifier of a most recent version of the metadata of the file, and for returning the most recent version of the metadata of the file to said each data processing device when the

comparison indicates that the metadata version identifier in ~~the~~ said each metadata request fails to identify the most recent version of the metadata of the file.

26. (Original) The file server as claimed in claim 25, wherein the version identifier is a number, and the data mover computer is programmed to increment the version identifier when the metadata of the file is modified.

27. (Original) A data processing system comprising, in combination;

a file server; and

a plurality of clients linked by a data network to the file server;

wherein the file server is programmed for receiving from each client at least one request for access to a file, for granting to said each client a lock on at least a portion of the file, and for sending to said each client metadata of the file including information specifying data storage locations in the file server for storing data of the file;

wherein said each client is programmed for using the metadata of the file to produce at least one data access command for accessing data of the file; and

wherein the file server is programmed for receiving from said each client said at least one data access command for accessing data of the file by accessing the data storage locations in the file server.

28. (Currently amended) The data processing system as claimed in claim 27, wherein the file server includes a data storage device including the data storage locations, and a data mover computer programmed for managing locks on files having data stored in said data storage device, wherein the data mover computer has a network port for receipt of file access requests ~~form~~ from clients, and wherein the data storage device has a network port for receipt of data access commands from said clients over at least one data transmission path that bypasses the data mover computer.

29. (Original) The data processing system as claimed in claim 28, wherein the data storage device stores metadata of a plurality of files having file data stored in the data storage device, the data mover computer is coupled to the data storage device for the transfer of the metadata between the data storage device and the data mover computer, the data mover computer has a random access memory, and the data mover computer is programmed for maintaining a metadata cache in the random access memory, and for accessing the metadata cache for obtaining the metadata that is sent to said each client.

30. (Original) The data processing system as claimed in claim 27, wherein a plurality of clients in the data network share read-write access to files stored in data storage of the file server, and the file server is programmed to grant respective read locks and write locks to the clients in the data network.

31. (Original) The data processing system as claimed in claim 27, wherein the lock on at least a portion of the file granted by the file server to the client is not granted to any particular application process of said each client, and wherein said each client has a lock manager for granting a local file lock to a particular application process that accesses the file.

32. (Original) The data processing system as claimed in claim 27, wherein said each client has a lock manager for responding to a request from an application process of said each client for access to the file by granting to the application process a local file lock on at least a portion of the file, and then sending to the file server said at least one request for access to the file.

33. (Original) The data processing system as claimed in claim 27, wherein said client is programmed with input-output related operating system routines for intercepting file access calls from client application processes for sending file access requests to the file server.

34. (Original) The data processing system as claimed in claim 33, wherein the data access commands include at least one write command for a write operation upon at least a portion of at least one file, and wherein the client is programmed for writing data to data storage locations in the file server, modifying the metadata from the file server in accordance with the write operation upon at least a portion of said at least one file, and sending the modified metadata to the file server.

35. (Original) The data processing system as claimed in claim 27, wherein said each client has a cache memory for caching the metadata of the file including a version identifier associated with the metadata of the file, and wherein said each client is programmed to include the version identifier in the request for access to the file, the file server is programmed to compare the version identifier from said each client to a version identifier of a most recent version of the metadata of the file, and the file server is programmed to return the most recent version of the metadata of the file to the client when the comparison of the version identifier from said each client to the version identifier of the most recent version of the metadata of the file indicates that the metadata of the file cached in the cache memory of said each client is not the most recent metadata of the file.

36. (Original) A program storage device containing a program for a file server, the file server having at least one data storage device for storing a file system, and having at least one network port for exchange of control information and metadata of files in the file system with at least one data processing device, the control information including metadata requests, wherein the program is executable by the file server for responding to each metadata request for metadata of a file by granting to said each data processing device a lock on at least a portion of the file, and returning to said each data processing device metadata of the file including information specifying data storage locations in the data storage device for storing data of the file.

37. (Original) The program storage device as claimed in claim 36, wherein the program is executable by the file server for receiving modified metadata from the data processing device and writing the modified metadata to data storage of the file server.

38. (Original) The program storage device as claimed in claim 36,

wherein the file server has nonvolatile data storage containing the file system and metadata of files in the file system, and the file server has a random access memory; and

wherein the program is executable by the file server for maintaining a metadata cache in the random access memory, and for accessing the metadata cache for obtaining the metadata that is returned to the data processing device.

39. (Original) The program storage device as claimed in claim 38, wherein the program is executable by the file server for receiving modified metadata from the data processing device, and writing the modified metadata to the metadata cache in the random access memory.

40. (Currently amended) The program storage device as claimed in claim 38, wherein the program is executable by the file server for receiving a metadata version identifier in ~~the~~ said each metadata request, for comparing the metadata version identifier in said each metadata request to a version identifier of a most recent version of the metadata of the file, and for returning the most recent version of the metadata of the file to the data processing device when the comparison indicates that the metadata version identifier in ~~the~~ said each metadata request fails to identify the most recent version of the metadata of the file.

41. (Original) The program storage device as claimed in claim 40, wherein the version identifier is a number, and the program is executable by the file server for incrementing the version identifier when the metadata of the file is modified.

42. (Original) A program storage device containing a program for a data processing device that is a client in a data network, the program being executable by the client to enable application programs of the client to access files in data storage of at least one file server in the data network, the program being executable in response to a call from an application program for access to data of a file by sending to the file server a metadata request for metadata of the file including information specifying data storage locations for data of the file in the file server, receiving the metadata of the file from the file server, using the metadata of the file to produce at least one data access command for accessing the data storage locations in the file server, and sending the data access command to the file server to access the data storage locations in the file server.

43. (Original) The program storage device as claimed in claim 42, wherein the program is executable by the client in response to the call from the application program by granting a local file lock to a particular application process that is executing the application program, and then sending to the file server the metadata request for metadata of the file.

44. (Original) The program storage device as claimed in claim 42, wherein the program includes input-output related operating system routines for intercepting file access calls from the client application programs.

45. (Original) The program storage device as claimed in claim 42, wherein the data access command is a write command for a write operation upon at least a portion of the file, and wherein the program is executable by the client for writing the data to the data storage locations in the file server, modifying the metadata from the file server in accordance with the write operation upon at least a portion of the file, and sending the modified metadata to the file server.

46. (Original) The program storage device as claimed in claim 45, wherein the program is executable by the client for sending the modified metadata to the file server after the client writes the data to the data storage of the file server.

47. (Original) The program storage device as claimed in claim 45, wherein the program is executable for sending the modified metadata to the file server in response to a commit request from the application program.

48. (Original) The program storage device as claimed in claim 45, wherein the program is executable for sending the modified metadata to the file server when the client requests the file server to close the file.

49. (Original) The program storage device as claimed in claim 42, wherein the client has a cache memory for caching the metadata of the file including a version identifier associated with the metadata of the file, and wherein the program is executable by the client for including the version identifier in the metadata request that is sent to the file server.

50. (Previously presented) A method of operating a file server and a client in a data network, the file server having a cached disk array including data storage locations, and a data mover computer for managing locks on files having data stored in the cached disk array, said method comprising:

(a) the client sending to the data mover computer at least one request for write access to a file;

(b) the data mover computer receiving said at least one request for write access to the file, granting to the client a lock on at least a portion of the file, and sending to the client metadata of the file including information specifying data storage locations in the cached disk array for storing data of the file;

(c) the client receiving from the data mover computer the metadata of the file, using the metadata of the file to produce at least one data access command for writing data to the data storage locations in the cached disk array for storing data of the file, the data access command including the data to be written to the data storage locations in the cached disk array for storing data of the file and specifying the data storage locations in the cached disk array for storing the data to be written, and sending the data access command over a data path that bypasses the data mover computer to access the data storage locations in the cached disk array for storing the data to be written;

(d) the file server responding to the data access command by writing the data to be written to the data storage locations in the cached disk array for storing the data to be written;

(e) the client modifying the metadata from the data mover computer in accordance with the writing of the data to be written to the data storage locations in the cached disk array for storing the data to be written; and

(f) the client sending the modified metadata to the data mover computer after the data has been written to the data storage locations in the cached disk array for storing the data to be written.